Information Sciences 418–419 (2017) 302–316

Contents lists available at ScienceDirect

Information Sciences

journal homepage: www.elsevier.com/locate/ins

An enhanced artificial neural network with a shuffled complex evolutionary global optimization with principal component analysis

Tiantian Yang[a,b,*], Ata Akabri Asanjan[a], Mohammad Faridzad[a], Negin Hayatbini[a], Xiaogang Gao[a], Soroosh Sorooshian[a]

[a] Center for Hydrometeorology and Remote Sensing (CHRS) & Department of Civil and Environmental Engineering, University of California, 5300 Engineering Hall (Bldg 308), Irvine, CA 92617, USA
[b] Deltares USA Inc. Silver Spring, MD, USA

ARTICLE INFO

Article history:
Received 18 October 2016
Revised 25 July 2017
Accepted 1 August 2017
Available online 4 August 2017

Keywords:
SP-UCI
Evolutionary algorithm
Artificial neural networks
Weight training
Global optimization

ABSTRACT

The classical Back-Propagation (BP) scheme with gradient-based optimization in training Artificial Neural Networks (ANNs) suffers from many drawbacks, such as the premature convergence, and the tendency of being trapped in local optimums. Therefore, as an alternative for the BP and gradient-based optimization schemes, various Evolutionary Algorithms (EAs), i.e., Particle Swarm Optimization (PSO), Genetic Algorithm (GA), Simulated Annealing (SA), and Differential Evolution (DE), have gained popularity in the field of ANN weight training. This study applied a new efficient and effective Shuffled Complex Evolutionary Global Optimization Algorithm with Principal Component Analysis – University of California Irvine (SP-UCI) to the weight training process of a three-layer feed-forward ANN. A large-scale numerical comparison is conducted among the SP-UCI-, PSO-, GA-, SA-, and DE-based ANNs on 17 benchmark, complex, and real-world datasets. Results show that SP-UCI-based ANN outperforms other EA-based ANNs in the context of convergence and generalization. Results suggest that the SP-UCI algorithm possesses good potential in support of the weight training of ANN in real-word problems. In addition, the suitability of different kinds of EAs on training ANN is discussed. The large-scale comparison experiments conducted in this paper are fundamental references for selecting proper ANN weight training algorithms in practice.

1. Introduction

The Artificial Neural Network (ANN) is a powerful, nonlinear, and adaptive mathematical predictive model that was inspired by the neurological structure of the human brain. According to the literature, the ANNs have been used extensively and successfully in various fields, including pattern recognition [16], image processing [1], ecological modeling [23], and water resources management [47], etc. During the development of ANNs, one of the significant advances is the implementation error BP concept [35]. In Rumelhart et al. [35], several neural networks were tested with the BP scheme, in which

* Corresponding author at: Center for Hydrometeorology and Remote Sensing (CHRS) & Department of Civil and Environmental Engineering, University of California, 5300 Engineering Hall (Bldg 308), Irvine, CA 92697, USA.
E-mail address: tiantiay@uci.edu (T. Yang).

http://dx.doi.org/10.1016/j.ins.2017.08.003
0020-0255/

the output-layer errors were purposely propagated into hidden-layers, and the optimal weights in the complete ANN were derived with gradient descent optimization. Furthermore, Rumelhart et al. [35] demonstrated that the BP scheme worked far faster than earlier approaches for training ANNs, and made it possible to use neural networks to solve problems that had been unsolvable in many fields. However, one of the drawbacks associated with BP and gradient-based optimizations is that the search tends to become stuck in local optimums, and optimization strategy lacks the capability to escape from local attractions. To optimization the ANN cost function is a complex, non-differentiable, and multi-modal problem. Therefore, the use of gradient-based optimizations is skeptical [40,44]. Furthermore, BP and gradient-based optimization schemes are extremely sensitive to initial conditions [18] and the prediction accuracy will dramatically decrease as the number of hidden neurons increases when using BP and gradient-based optimization schemes [14].

Given these aforementioned weaknesses of BP and gradient-based optimization schemes used in classical ANN training, during the last decade, many researchers and model developers have been attempting to use different types of EAs, such as GA, PSO, SA, and DE, as the alternatives for the BP in the ANN weight training process. According to the literature, Ding et al. [5] reviewed the many uses of EAs in optimizing ANNs weights, and pointed out that the BP algorithm appeared to be more effective when used in local searches, while the GA was good for global searches. Ilonen et al. [15] compared a DE algorithm against the classical gradient-based methods in the training process of a Feed-Forward Neural Network and concluded that the optimal weights found by DE were always equal to or better than the initial optimal weights found by the gradient-based methods when the computational time was relaxed. Gudise and Venayagamoorthy [12] compared the computational efficiency of ANNs using PSO and BP in learning a non-linear function, and they proved that PSO was a faster-learning algorithm than the BP scheme. Jain and Srinivasulu [17] employed a GA in ANN rainfall-runoff models and proved that the ANN trained with the GA was able to obtain more accurate, low-magnitude flow simulation than the ANN trained with the BP scheme in the Kentucky River watershed. Slowik and Bialko [37] used multiple methods to train an ANN, including the DE algorithm, BP, and a Levenberg–Marquardt method. They found that the DE-based ANN could obtain better classification in the presumed time than the other two methods. Dai et al. [4] investigated a number of EAs in training the ANN, and concluded that all of the heuristic search algorithms were superior over BP with regard to the regulation performances, but less computationally efficient. Sulistiyo and Dayawati [39] developed a GA-type evolutionary strategy in optimizing the weights of a Feed-Forward ANN and determined that the EA-based ANN had similar performances in the training phase when compared to the BP-based ANN, but yielded consistently smaller errors in the testing phase.

According to the literatures mentioned above, it has been widely acknowledged that the BP and gradient-descent optimization schemes have many drawbacks, and that the EAs are proven to be useful in remedying the associated weaknesses with BP and gradient-based optimizations. However, given a real-world regression or classification problem, selecting a proper ANN training algorithm is a tedious task, which requires a large number of independent experiments and inclusive comparisons among all kinds of EAs. Most of the comparison experiments summarized above were limited to a few particular cases and comparison studies were carried out comparing a proposed EA against the BP scheme. In order to get a more comprehensive comparison among all types of EAs in assisting ANN weight training, in this study, one of the focuses is to carry out a large-scale numerical experiment and test the performances of different EAs with regards to their enhancements on optimizing ANN training weights. The experiments in this paper include 17 benchmark, real-world, machine-learning datasets retrieved from the UCI computer science repository [24].

In addition, another significance of this study is to introduce a powerful evolutionary optimization scheme, termed the SP-UCI [2] in support of the ANN weight training. The SP-UCI algorithm is a population-based, global, evolutionary search scheme, which was developed based on an efficient and effective Shuffled Complex Evolutionary Global Optimization – University of Arizona (SCE-UA) scheme [6]. The family of the SCE-UA algorithm includes various versions, which were developed for different tasks, such as the multi-objective optimization [46], distribution estimations in a Bayesian framework, and high-dimensional optimization problems [2,3]. All kinds of SCE-UA descendants are proven to be effective and efficient in the fields of optimization, computer sciences, hydrology, and water resources management [2,3,6,46]. According to Chu et al. [2,3], SP-UCI combines the strengths of (1) the Nelder–Mead simplex downhill local search scheme [30], (2) the global complex competition evolution [6], and (3) the Principal Component Analysis (PCA) [19]. The advantage of the SP-UCI lies in its capability to address the high-dimensional challenge, or termed as the “curse of dimensionality”, which is commonly associated with complex, real-world problems. With respect to the ANN weight training problem, the total number of ANN connectivity weights between input-hidden layers, and hidden-output layers is quite large, which makes the optimization of ANN weight a suitable problem for applying the SP-UCI algorithm.

The superiority of the SP-UCI over GA, PSO, and DE on composite test functions has already been demonstrated in Yang et al. [46], however, no study has been conducted to implement the SP-UCI in ANN training and investigate its suitability in tuning the ANN weights. In another previous study conducted by Gupta et al. [13], the Nelder–Mead simplex downhill scheme, which was used in SP-UCI, has been tested against BP and the conjugated gradient-descent scheme with regard to training the ANN weights on a number of simple test functions. According to the experiments demonstrated in Gupta et al. [13], the simplex downhill scheme is capable of producing residuals similar to those of BP and the conjugated gradient-descent scheme, but requires fewer function evaluations. In addition, Gupta et al. [13] concluded that when using a global search procedure, i.e., the multi-start simplex approach, the associated risk of population converging to a sub-optimal solution could be reduced. Nevertheless, the SP-UCI algorithm remained untested, although the suitability of the simplex downhill scheme in training the ANN and the sensitivity of the initial start position in the weight space were discussed in Gupta et al. [13]. Given the fact that the Nelder–Mead simplex downhill scheme is only one of the three core-technics that con-

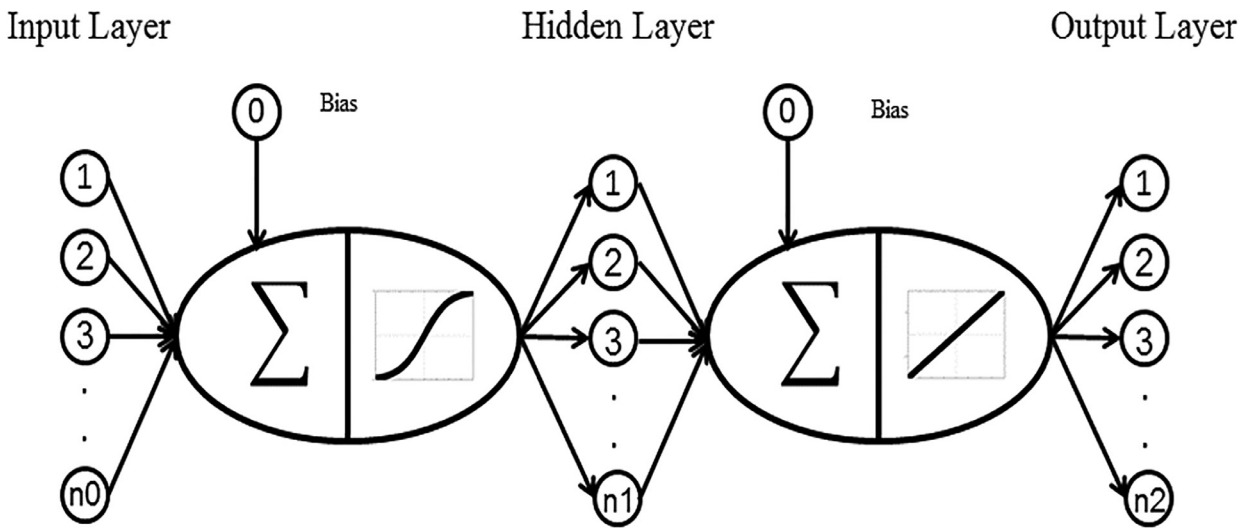

Fig. 1. A Three-Layer Feed-Forward Neural Network (TLFFNN).

stitute the SP-UCI algorithm, a direct implementation of the SP-UCI algorithm into ANN weight training and comparison against other EAs needs to be carried out.

Despite using EAs in training an ANN with a specific architecture or topography, the use of EAs in the optimal design of an ANN structure is another popular research topic in the literature. According to Yao [45], to design an ANN architecture is a trial-and-error process and is always subject to users' preferences and expert experiences. EAs are also found to be useful tools in automatically designing ANN structures and topographies as summarized by Yao [45], such as the number of hidden neurons, layer connectivity, and the transfer function of each neuron, etc. Nevertheless, Fogel [9] argued that the evolution of architectures without any weight training would not give accurate fitness, which indicated the importance of proper tuning of ANN weights. A remedy will be to use EAs on ANN architecture and weight training simultaneously and collaboratively so that a higher accuracy can be reached. As concluded by Yao [45], simultaneous applying EAs on different levels of ANN, i.e., architecture, weight-space training and learning rules, can be inefficient and unnecessary due to the large number of function evaluations required by various EAs.

Therefore, in this study, we present a pioneer work of applying the SP-UCI algorithm in training the ANN weights and comparing its performances with multiple state-of-the-art heuristic search methods, including GA, PSO, SA, and DE. The main contributions of this study are (1) introduce a new type of EA, termed the SP-UCI algorithm, into the weight training process of an ANN model, and (2) compare and investigate the suitability of different kinds of heuristic search optimizations on large-scale, multi-disciplinary, benchmark machine-learning datasets. The numerical comparison results will provide useful information and references for any future study for choosing proper EAs as ANN training algorithms.

The organization of this paper is as follows: Section 2 introduces the methodologies, including ANN and employed heuristic search methods (SP-UCI, GA, PSO, SA and DE), and the benchmark machine-learning datasets. Section 3 presents the experiment results; Discussion is provided in Section 4, and Section 5 summarizes the main finding, conclusions and future works.

2. Methodology and data

2.1. ANN and BP

The ANN is a powerful predictive model initially introduced by McCulloch and Pitts [29], which mimics the neurological structure of the human brain [25]. In the structure of the ANN, a non-linear regression mode of target variables is built on decision variables (also called predictors or features). The hearts of an ANN structure are: (1) its hidden layers that consist of numbers of hidden nodes, and (2) the activation functions processing and extracting explicit information between features and target variables. The ANN is used extensively in many fields of study, such as pattern recognition [16], image processing [1], ecological modeling [23], and water resources management [6,28,47,48], etc. In the literature, one of the most commonly used ANN structures is a Three-Layer Feed-Forward Neural Network (TLFFNN) [41,47] (Fig. 1). As mentioned in Pratt [33], when the number of hidden nodes increases, the weight-space response surface will become much rougher, which jeopardizes the global convergence of many classical gradient-based optimization schemes. Therefore, the number of hidden nodes in the TLFFNN is set to 10, which results in a high-dimensional weight space for our selected benchmark datasets.

As shown in Fig. 1, a typical TLFFNN has three major parts: an input layer $\vec{x}(x_1, x_2, \ldots, x_{n_0})$, a hidden layer $\vec{h}(h_1, h_2, \ldots, h_{n_1})$, and an output layer $\vec{z}(z_1, z_2, \ldots, z_{n_2})$, where n_0, n_1, and n_2 represent the total number of inputs, hidden neurons, and outputs, respectively. A transformation function (f) is used to connect the input with hidden neurons, and it is commonly calculated as the weighted sum of inputs, as shown in Eq. (1).

$$\mathrm{h}_j = f\left(\sum_{i=1}^{n_0} w_{ij}x_i + w_{0j}\right) \tag{1}$$

where h_j is the jth neuron in the hidden layer, x_i is the ith input, w_{ij} represents the weight assigned to the ith input in order to calculate the jth hidden neuron, and $\mathrm{j} \in (1, 2, \ldots n_1)$ and $\mathrm{i} \in (1, 2, \ldots n_0)$.

Following a similar approach, another transformation function (g) is used to connect hidden neurons to the outputs, which is shown in Eq. (2).

$$\mathrm{z}_k = g\left(\sum_{j=1}^{n_1} v_{jk}h_j + v_{0k}\right) \tag{2}$$

where z_k is the kth value in the output layer, h_j is the jth neuron in the hidden layer, v_{jk} represents the weight assigned to the jth neuron in the hidden layer, and $\mathrm{k} \in (1, 2, \ldots n_2)$ and $\mathrm{j} \in (1, 2, \ldots n_1)$. In any ANN, the transformation functions f and g that connect each layer are called connectivity functions or activation functions. In this study, a hyperbolic tangent transformation function (Eq. (3)) and a linear transformation function (Eq. (4)) are used to connect input layer to hidden layer, and connect hidden layer to output layer, respectively.

$$h_j = \frac{2}{1 + e^{-2\sum_{i=1}^{n_0} w_{ij}x_i + w_{0j}}} - 1, \ \mathrm{j} = 1, 2, \ldots \quad n_1 \tag{3}$$

$$z_k = \sum_{j=1}^{n_1} v_{jk}h_j + v_{0k}, \ \mathrm{k} = 1, 2, \ldots \quad n_2 \tag{4}$$

The weights w_{ij} and v_{jk} are obtained by minimizing the Sum of Square Errors (SSE) between model output and the target variable (Eq. (5)). To minimize Eq. (5) becomes an optimization problem, whereas, in this study, multiple EAs are employed and tested, including the proposed SP-UCI, GA, PSO, SA, and DE.

$$\mathrm{argmin(SSE)} = \mathrm{argmin}\left[\frac{1}{2}\sum_{k=1}^{n_2}(t_k - z_k)^2\right] \tag{5}$$

2.2. Heuristic search optimization

2.2.1. SP-UCI

The Shuffled Complex Evolution with Principal Component Analysis – University of California, Irvine (SP-UCI) is first introduced by Chu et al. [2]. The SP-UCI is based on an effective and efficient Shuffled Complex Evolution scheme - University of Arizona (SCE-UA) [6]. Since the debuts of SP-UCI and SCE-UA, both of them they have been demonstrated as superior or competitive optimizers in comparison with other heuristic search schemes, such as GA, PSO, SA, and DE optimizations based on the their performance on both composite test functions [46] and many real-world applications [46,49].

The difference between SP-UCI algorithm and its predecessor, SCE-UA algorithm, is the introduction of Principal Component Analysis (PCA) with the purpose of monitoring and maintaining the population diversity during the entire evolution process [2]. Due to the roughness of the response surface and the complexity of high-dimensional optimization problems, when offspring particles converge to a subspace within the original parameter space, the search will be restricted in the subspace instead of a full span of the parameters of a problem. This phenomenon is commonly referred to "population degeneration", which prevents the global convergence of many direct-search algorithms [2]. By recursively using the PCA technique during the evolutions of SCE-UA, the simplexes are allowed to search the full parameter space instead of collapsed into a subspace; therefore, the risks of population premature convergence and becoming trapped in local optimal are avoided.

To provide more detail, the procedure in SP-UCI includes the following steps: (1) The algorithm randomly samples m $\times$ p points in the search space as the initial population, where m is the number of complexes, and p is the number of individuals in a complex. (2) The entire population is randomly shuffled and split into m complexes. (3) The individuals in each complex are sorted based on their fitness, i.e., the objective function values. (4) A simplex, i.e., a concave object with n $+1$ vertices, is constructed in each complex using a triangular possibility function, where n is the dimensionality of the problem. (5) A Nelder–Mead simplex downhill optimization scheme [30] is carried out for each constructed simplex independently. (6) When reaching a user-defined number of iterations, all individuals in each complex are transformed into an orthogonal coordinate system, and the diversity and dimensionality are checked for each Principal Component (PC). (7) If any "population degeneration" phenomenon is identified in a given complex, i.e., a relative small standard deviation of population along each PC, a multi-normal resampling is executed with regard to the PCs that produce relatively small standard

deviation. (8) After the resampling, the entire individuals in the orthogonal coordinate system are updated and transformed back into the ordinary coordinate system to perform the Nelder–Mead simplex-downhill evolution for the next loop. (9) The procedure (2)–(8) is repeated and looped until stopping criteria are met, i.e., the maximum number of iterations is reached or population convergence is fulfilled. For interested readers, the algorithm flowcharts, detailed information and mathematical descriptions are available in Chu et al. [2] and Yang, et al. [46].

2.2.2. Other heuristic search methods

The Genetic Algorithm (GA) belongs to one of the most popular evolutionary algorithms that mimic the processes of natural selection [10]. Natural selection is defined as the processes that organisms correspondingly survive and then produce offspring who consistently process the tendency to adapt their environment. There are different types of natural selection processes, including chromosome heredity, mutation, crossover and selection.

According to Simpson, et al. [36], the optimization of a particular problem using GA is achieved through the following procedures: (1) Randomly select sample a number of individuals to form the initial population. (2) All individuals are evaluated using the objective function and scored as fitness values. (3) Select a number of members as parents and those individuals with lower fitness values are selected as elite members. (4) The parent members produce their offspring using mutation and crossover, while these elite members are passed to the next population without any changes. (5) Replace the current population with the offspring and elite members. (6) Repeat Steps 2–5 until the stopping criteria are met, such as the average relative changes in the fitness of functions during last iterations, or the user defined maximum number of function evaluation. The GA code used in this study is a real-value coded version from the Matlab global optimization package, which is one of the well-developed and stable GA toolboxes. The GA algorithm has been used in optimizing the ANN connectivity weights and proven to be useful and efficient [42].

Similar to GA, the Particle Swarm Optimization (PSO) is another extensively used, population-based global optimizer, which simulates the social-individual behaviors of bird flocking and fish schooling [20,21]. Instead of natural selection operators, i.e., mutation or crossover, in PSO, the offspring production is based on the fitness of individuals (particles) and their movement velocities towards the individual that has the best fitness value. This is a simplified mimic of social behavior of bird foraging, in which the search mechanism has been proven to be efficient and effective in Eberhart and Kennedy [7]. According to Eberhart and Kennedy [7], it is assumed in the PSO search mechanism that all birds (individuals) are unaware of food sources (global optimum); therefore, one of the effective foraging strategies for bird flock is to fly toward the bird which is nearest to the food. It worth mentioning that the search mechanism in PSO is different from that in GA. The population in PSO is updated by approaching two best positions: (1) the best location that gives the best fitness value within the neighborhoods of current positions of all individuals, and (2) the historical best location that gives the best fitness value throughout the entire evolution that each individual has achieved so far, while in GA the individuals move as a group approaching the global optimum [31]. The employed PSO code is obtained from the standard Matlab global optimization toolbox.

The Simulated Annealing (SA) algorithm was originally introduced by Kirkpatrick [22] as a robust global optimizer for addressing the issue of trapping in local minimums of the classical gradient-descent method. The concept of SA was inspired by the process of annealing in metal-work, in which a metal material was repeatedly heated and cooled down to improve the stiffness of metals. In the metal-work process, metal is heated to a pre-defined temperature, which will allow the metal molecules to vibrate in their neighborhood, and partially break the molecular bonds. Followed by the heating process, a cooling process reforms the molecular structure and recombines stronger molecular bonds in a way that the whole physical system reaches an entropy maximum state.

This metal-work annealing concept can be used creatively for In Simulated Annealing optimization, a high temperature is used as the reheating threshold, which gradually decreases during the evolution. With a higher reheating-temperature, the algorithm is allowed to accept any solution that is worse than the current best with a higher frequency. As the reheating-temperature threshold decreases as the evolution proceeds, the algorithm is gradually allowed to focus only on searching a limited neighborhood of best solutions with reduced chances to accept worse solutions. As the reheating-temperature decreases, the chances of accepting worse solutions will decrease. As a result, the search will converge after a number of user-defined function evaluation is reached [11]. The SA algorithm has been proven to be effective in finding global optima on multi-modality response surfaces and many real-world problems [8]. The used SA code is from the Matlab global optimization package with default reheating-temperature and tolerance settings.

The Differential Evolution (DE) algorithm is a global, evolutionary optimization algorithm which is similar to GA and originally coined by Storn and Price [38]. The DE algorithm uses Darwin's natural selection theory of mutation, crossover, and selection to produce better candidates for "survival" in the scope of fitness values. During the evolution of the DE algorithm, a mutation process is first employed to produce a mutated offspring by adding a scaled difference between two randomly selected vectors, or individuals, to the corresponding members in the population, called donor vectors. Then, a trial offspring, or trial vector, is created by carrying the crossover of randomly selected parent vectors or individuals. Finally, the mutated offspring and the trial offspring are compared and the one with better fitness value is used to update the population [34]. The difference between DE and GA is that, in GA, the operations of crossover and mutation of chromosomes are performed simultaneously as a group-mating process while, in DE, the crossover and mutation jointly work as a competitive procedure to generate offspring. The DE code used in this study was obtained from Dr. Wei Chu, which was the one used in

Table 1
Detailed information on the selected benchmark datasets from UCI machine learning repository.

Dataset No.	Name	No. features	Target variable type	Weight space dimension	Population size
1	Airfoil Self-Noise	5	Real	64	258
2	Auto MPG	7	Real	82	330
3	Automobile	22	Real	217	870
4	Breast Cancer Wisconsin (Prognostic)	10	Integer	109	438
5	Challenger USA Space Shuttle O-Ring	3	Integer	46	186
6	Combined Cycle Power Plant	4	Real	55	222
7	Computer Hardware	9	Integer	100	330
8	Concrete Slump Test (Strength)	7	Real	82	330
9	Concrete Slump Test (Slump)	7	Real	82	330
10	Concrete Slump Test (Flow)	7	Real	82	330
11	Concrete Compressive Strength	8	Real	91	366
12	Fertility	9	Integer	100	402
13	Forest Fires	10	Real	109	438
14	Housing	13	Real	136	546
15	ISTANBUL STOCK EXCHANGE	7	Real	82	330
16	Energy efficiency (Heating Load)	8	Real	91	366
17	Energy efficiency (Cooling Load)	8	Real	91	366

his previous published studies [2]. Some recent developments and applications of DE algorithm are available at Poikolainen et al. [32].

2.3. Datasets and setting

2.3.1. Datasets

In order to test the suitability of the proposed SP-UCI algorithm and investigate the suitability of different kinds of EA-enhanced ANNs, we aggressively carried out a large-scale comparison over 17 benchmark and real-world datasets selected from different fields. The datasets were retrieved from the UCI machine-learning datasets repository (https://archive.ics.uci.edu/ml/datasets.html) [24], which has been used extensively in numerous model- and algorithm-evaluation studies. The types of datasets used in this study are all multivariable and regression. The scope of selected datasets covers life, engineering, social sciences, and business. Table 1 lists the dataset name, number of features, target variable type, dimension of the weight space, and the size of population for the selected datasets. The dimension of weight space depends on the number of features in each dataset, which is shown in Eq. (6). In order to produce a fair comparison, the population size is set to identical for all algorithms, which follows the default setting in the SP-UCI algorithm, as shown in Eq. (7). As the number of features and complexity of databases increase, the population size correspondingly increases, which allows all algorithms to have enough numbers of sampling in the weight space at the beginning of the searching.

$$\text{Dimension} = (\text{Number of Features} + 1) \times 9 + 10 \tag{6}$$

$$\text{Size of Population} = (2 \times \text{Dimension} + 1) \times 2 \tag{7}$$

2.3.2. Experiment design

In all performed experiments, 70% of the data is used for training the ANN model, 15% of the data is used for validation, and the remaining 15% is held out as an independent testing dataset. To examine the reproducibility of the comparison experiments, we first conducted 30 independent runs of each algorithm using an identical split of training, validation, and testing datasets. Furthermore, another 30 runs were carried out for each algorithm using randomly shuffled training, validation, and testing datasets in order to test the predictive performances of the models using different training datasets. In other words, there is a total of number of 60 runs performed for each algorithm. 30 of the 60 runs use identical training, validation, and testing datasets to test the reproducibility of the experiment results. In another 30 runs, before executing optimization algorithms, the datasets are shuffled and split into training, validation, and testing datasets to test the reliability of algorithm performances. Last, all five EA optimization schemes (SP-UCI, PSO, GA, SA, and DE) are implemented to the TLFFNN, and the convergence performances are compared in the training, validation, and testing datasets, respectively.

The crossover and migration fraction rates used in GA are 0.8 and 0.2 respectively. A Gaussian mutation scheme is used with default value of 1 for both scale and shrink rates. By default setting for PSO, the cognitive and social attraction rates are 0.5 and 1.25, respectively. The initial temperature in SA is 100 for each dimension. By default, the reheating temperature to the initial temperature ratio equals to 0.95 to the power of iteration number. The crossover probability of 0.7 is used in DE. A default value of 2 is set to both the number of simplexes in a complex, and total number of complexes in SP-UCI.

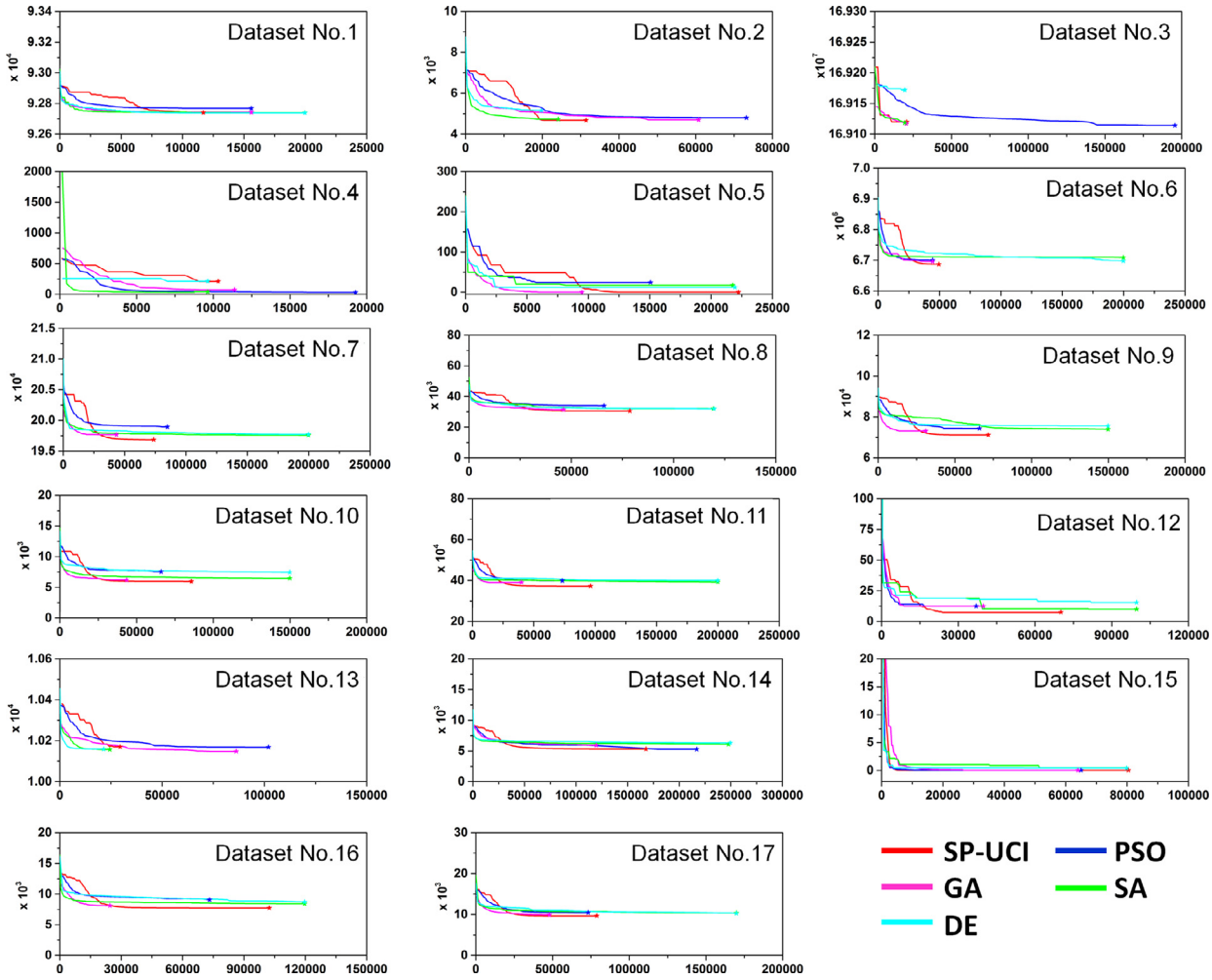

Fig. 2. The evolution processes of SP-UCI-, GA-, PSO-, SA-, and DE-based ANN on different datasets.

3. Results

3.1. Training and validating period

As mentioned above, there are a total of 30 independent runs performed for each algorithm using the same training datasets. In Fig. 2, the evolution processes of SP-UCI-, GA-, PSO-, SA-, and DE-based ANN for one single run randomly selected among the 30 independent runs are shown, where the x-axis indicates the number of function evaluations, and the *y*-axis represents the Sum of Square Errors (SSE) as objective function values. The final converged objective function value for each algorithm is marked with colored stars for each algorithm. A lower final-objective function value (smaller y-value) indicates a better convergence performance. A smaller number of function evaluations (smaller x-value) means a higher computational efficiency of the search algorithms. As shown in Fig. 2, the search starts with relatively high objective function values (error terms), and the objective function values decrease as the search evolves. To quantitatively compare the ANN performances of the demonstrated run results in Fig. 2, the final objective function values for training, validation, and testing datasets are shown in Table 2. In Table 2, the bold and underlined values for each row indicate the best final objective function values, i.e., the smallest errors, achieved by all EAs on each database. The smaller the values in the training, validation, and testing datasets, the better the convergence performances with the performed EAs.

In addition, we conducted 30 independently repeated runs to examine the reproducibility of the demonstrated run results. The statistics of the final objective function values for each algorithm are calculated and shown in Table 3. The calculated statistics include the mean of the final objective function values for all 30 independent runs, and the Standard Deviation (Std). The lower the mean value, the better convergence an algorithm can produce. The smaller the Std value, the better the consistency of reproducing similar results. Similarly, the bold and underlined values for each row in Table 3 indicate the best (smallest) mean and Std values for each database.

Table 2
The statistical performances of SP-UCI-, PSO-, GA-, SA-, and DE-based ANN on validation and testing datasets (bold and underlined values indicate the best statistics for each database).

Dataset No.	SP-UCI			PSO			GA			SA			DE		
	Train	Val.	Test	Train	Val.	Test	Train	Val.	Test	Train	Val.	Test	Train	Val.	Test
1	9.274e4	2.037e4	4.076e4	9.277e4	2.037e4	4.077e4	**9.274e4**	2.037e4	**4.076e4**	9.274e4	**2.037e4**	4.076e4	9.274e4	2.037e4	4.076e4
2	**4.681e3**	**9.171e2**	1.925e3	4.801e3	9.404e2	1.975e3	4.700e3	9.192e2	**1.921e3**	4.795e3	9.499e2	1.955e3	5.152e3	1.024e3	2.144e3
3	1.644e8	6.155e7	7.972e7	1.644e8	3.916e7	7.969e7	**1.644e8**	**3.916e7**	7.969e7	1.644e8	3.916e7	**7.969e7**	1.644e8	3.916e7	7.969e7
4	2.117e2	1.754e1	1.205e2	**2.128e1**	**3.150e0**	**1.521e1**	7.179e1	1.603e1	5.686e1	4.059e1	9.198e0	2.121e1	2.148e2	5.469e1	5.374e1
5	**0.000e0**	5.000e−1	1.000e0	2.450e1	4.500e0	1.300e1	**0.000e0**	**0.000e0**	**0.000e0**	1.750e1	5.500e0	1.200e1	1.250e1	5.000e−1	2.000e0
6	**6.687e6**	**1.431e6**	**2.863e6**	6.700e6	1.433e6	2.869e6	6.696e6	1.432e6	2.873e6	6.709e6	1.435e6	2.872e6	6.698e6	1.433e6	2.868e6
7	**1.968e5**	**3.057e4**	**6.118e4**	1.989e5	3.106e4	6.217e4	1.977e5	3.078e4	6.154e4	1.976e5	3.076e4	6.155e4	1.977e5	3.075e4	6.155e4
8	**3.064e4**	**6.379e3**	**1.331e4**	3.397e4	6.969e3	1.457e4	3.271e4	6.768e3	1.372e4	3.205e4	6.612e3	1.383e4	3.198e4	6.586e3	1.374e4
9	**5.972e3**	**1.582e3**	**3.411e3**	7.536e3	1.982e3	4.276e3	7.091e3	1.982e3	3.552e3	6.463e3	1.879e3	3.691e3	7.461e3	1.968e3	4.228e3
10	**7.118e4**	**1.511e4**	**3.231e4**	7.436e4	1.573e4	3.379e4	7.312e4	1.608e4	3.316e4	7.397e4	1.569e4	3.377e4	7.558e4	1.599e4	3.426e4
11	**3.728e5**	**7.226e4**	**1.481e5**	3.993e5	7.786e4	1.594e5	4.101e5	7.997e4	1.542e5	3.929e5	7.648e4	1.567e5	3.998e5	7.789e4	1.595e5
12	**7.500e0**	**3.000e0**	**6.000e0**	1.250e1	5.500e0	1.100e1	1.250e1	3.000e0	7.000e0	1.000e1	3.500e0	7.000e0	1.550e1	7.000e0	1.400e1
13	1.017e4	3.377e2	6.764e2	1.017e4	3.344e2	6.701e2	**1.015e4**	3.318e2	**6.605e2**	1.016e4	**3.311e2**	6.611e2	1.016e4	3.311e2	6.638e2
14	5.327e3	1.316e3	**2.659e3**	**5.292e3**	**1.308e3**	3.015e3	5.925e3	1.455e3	2.942e3	6.082e3	1.515e3	3.060e3	6.304e3	1.561e3	3.157e3
15	**2.947e−2**	**5.173e−3**	**1.051e−2**	3.213e−2	5.231e−3	1.068e−2	3.158e−2	5.100e−3	8.059e−2	4.487e−1	1.084e−1	2.178e−1	4.318e−1	9.219e−2	1.864e−1
16	**7.737e3**	**1.751e3**	**3.502e3**	9.069e3	2.049e3	4.099e3	9.225e3	2.077e3	3.670e3	8.422e3	1.904e3	3.808e3	8.697e3	1.952e3	3.903e3
17	**9.674e3**	**2.089e3**	**4.250e3**	1.045e4	2.255e3	4.590e3	1.079e4	2.328e3	4.356e3	1.039e4	2.237e3	4.550e3	1.029e4	2.214e3	4.501e3

Table 3
Statistical performances of SP-UCI-, PSO-, GA-, SA-, and DE-based ANN on same training datasets with 30 independent runs (bold and underlined values indicate the best statistics for each dataset).

Dataset No.	SP-UCI		PSO		GA		SA		DE	
	Mean	Std	Mean	Std	Mean	Std	Mean	Std	Mean	Std
1	9.276e4	2.330e1	9.276e4	1.824e1	9.275e4	**3.185e1**	**9.274e4**	8.890e1	9.282e4	6.361e1
2	5.209e3	4.456e2	4.746e3	2.162e2	**4.709e3**	1.016e2	4.885e3	2.082e2	6.452e3	**9.893e1**
3	1.644e8	2.861e3	1.644e8	6.585e3	**1.644e8**	**1.740e3**	1.644e8	5.780e3	1.644e8	1.748e3
4	2.005e2	7.283e1	**2.012e1**	**1.179e0**	5.951e1	8.910e0	3.475e1	3.609e0	2.157e2	2.963e1
5	**0.000e0**	**0.000e0**	2.322e1	1.726e1	0.000e0	0.000e0	1.517e1	4.496e0	3.900e1	6.282e0
6	**6.687e6**	**3.759e2**	6.719e6	1.444e2	6.712e6	4.427e2	6.705e6	8.106e2	6.777e6	5.497e2
7	**1.968e5**	**1.461e2**	1.983e5	5.920e2	1.976e5	1.431e2	1.979e5	3.127e2	2.017e5	2.162e2
8	**3.064e4**	**3.267e0**	3.334e4	1.007e3	3.183e4	4.309e2	3.265e4	6.008e2	3.920e4	4.059e2
9	**7.119e4**	**6.991e0**	7.540e4	1.232e3	7.302e4	5.268e2	7.432e4	7.268e2	8.328e4	4.709e2
10	**5.972e3**	**1.485e0**	6.947e3	3.235e2	6.274e3	9.549e1	6.449e3	2.654e2	9.421e3	1.849e2
11	**3.728e5**	**7.086e1**	4.032e5	8.860e3	3.878e5	3.818e3	3.962e5	5.490e3	4.592e5	3.071e3
12	**7.333e0**	**1.011e0**	1.082e1	2.365e0	9.600e0	2.444e0	1.557e1	1.837e0	2.167e1	1.184e0
13	1.025e4	3.709e1	1.016e4	1.201e1	**1.015e4**	**3.405e1**	1.017e4	1.003e1	1.031e4	8.998e1
14	**5.295e3**	**1.611e1**	6.299e3	1.751e2	5.802e3	1.444e2	6.148e3	1.323e2	7.819e3	6.950e1
15	**2.891e−2**	**5.370e−4**	3.555e−2	5.841e−3	1.719e−1	9.424e−2	4.561e−1	1.146e−1	4.977e−1	2.009e−1
16	**7.740e3**	**2.162e0**	8.673e3	4.176e2	8.130e3	1.703e2	8.573e3	2.580e2	1.132e4	1.227e2
17	**9.674e3**	**1.936e0**	1.085e4	4.106e2	1.006e4	1.422e2	1.055e4	2.295e2	1.367e4	1.426e2

Table 4
Statistical performances of SP-UCI-, PSO-, GA-, SA-, and DE-based ANN on 30 randomly shuffled training datasets (bold and underlined values indicate the best statistics for each dataset).

Dataset No.	SP-UCI		PSO		GA		SA		DE	
	Mean	Std	Mean	Std	Mean	Std	Mean	Std	Mean	Std
1	9.430e4	**4.012e3**	9.430e4	4.013e3	9.429e4	4.015e3	**9.428e4**	4.013e3	9.436e4	4.015e3
2	5.048e3	3.014e2	4.813e3	2.364e2	**4.764e3**	**1.587e2**	4.879e3	2.347e2	6.477e3	1.669e2
3	1.654e8	1.039e7	**1.653e8**	**1.039e7**	1.653e8	1.039e7	1.653e8	1.039e7	1.654e8	1.039e7
4	2.053e2	7.311e1	**1.917e1**	**3.105e0**	6.187e1	6.893e0	3.299e1	3.701e0	2.358e2	3.694e1
5	**0.000e0**	**0.000e0**	2.082e1	1.625e1	3.333e1	1.269e−1	1.433e1	4.346e0	4.027e1	5.338e0
6	**6.687e6**	**3.936e3**	6.718e6	1.168e4	6.712e6	6.154e3	6.707e6	9.431e3	6.778e6	5.365e3
7	**2.392e5**	**5.238e4**	2.405e5	5.257e4	2.399e5	5.245e4	2.402e5	5.248e4	2.442e5	5.272e4
8	**3.074e4**	**9.256e2**	3.374e4	1.336e3	3.191e4	1.040e3	3.297e4	1.263e3	3.930e4	1.282e3
9	**7.321e4**	**3.355e3**	7.712e4	3.469e3	7.511e4	3.506e3	7.656e4	3.405e3	8.541e4	3.848e3
10	**6.126e3**	**3.306e2**	7.196e3	4.599e2	6.458e3	3.947e2	6.561e3	3.914e2	9.659e3	4.943e2
11	**3.759e5**	**8.848e3**	4.051e5	1.286e4	3.890e5	9.930e3	3.991e5	1.168e4	4.621e5	9.634e3
12	**1.440e0**	**1.030e1**	3.253e0	8.650e0	2.649e0	1.477e1	2.250e0	2.067e1	3.174e0	1.440e0
13	7.174e3	3.299e3	7.112e3	3.277e3	**7.103e3**	**3.272e3**	7.119e3	3.278e3	7.228e3	3.304e3
14	**5.121e3**	**2.166e2**	6.102e3	3.157e2	5.607e3	2.781e2	5.868e3	2.598e2	7.589e3	3.029e2
15	**2.992e−2**	**1.520e−3**	4.042e−2	1.222e−2	1.964e−1	2.016e−1	4.444e−1	1.910e−1	5.226e−1	1.695e−1
16	**7.985e3**	2.011e2	9.104e3	3.787e2	8.324e3	**1.904e2**	8.841e3	3.501e2	1.156e4	2.746e2
17	**9.687e3**	**1.929e2**	1.101e4	4.565e2	1.012e4	2.740e2	1.057e4	3.876e2	1.364e4	2.506e2

3.2. Testing period

Using another 30 randomly shuffled datasets, the statistics of the final objective function values for each algorithm are calculated and shown in Table 4. The difference between the 30 independent runs and the 30 shuffled runs is that, under the scenario of independent runs, the splits of training, validation, and testing datasets are kept identical for all 30 runs, whereas the datasets partitions are different among all 30 runs under the shuffled scenario. The tests on 30 shuffled datasets are intended to investigate whether the performances of algorithms are consistent if different combinations of data points in a particular dataset are used, which is a common approach in the field of machine-learning. It is possible that one algorithm can perform better than others on a specific combination of training data points, while the performance cannot be achievable if different training data points are used.

When comparing the best statistics (bold and underlined values) in Tables 3 and 4, there are some differences with regard to the final convergence values, and the best performed algorithm class. This is because the random shuffling of data points is able to construct different training databases, and to form different regression modes for each EA. When the ANN connectivity weights are trained using different training datasets, the performances in training, validation, and testing datasets vary from one shuffling combination to another.

Table 5
The statistical performances of SP-UCI-, PSO-, GA-, SA-, and DE-based ANN on testing datasets with 30 independent runs (bold and underlined values indicate the best statistics for each dataset).

Dataset No.	SP-UCI		PSO		GA		SA		DE	
	Mean	Std	Mean	Std	Mean	Std	Mean	Std	Mean	Std
1	4.077e4	1.006e2	4.077e4	7.877e2	**4.076e4**	**1.375e2**	4.076e4	3.838e2	4.079e4	2.724e2
2	2.163e3	1.942e2	1.958e3	9.402e1	**1.946e3**	4.604e1	2.024e3	9.280e1	2.688e3	**3.678e1**
3	7.971e7	1.669e3	**7.969e7**	3.160e3	7.969e7	**1.477e3**	7.969e7	3.359e3	7.971e7	1.507e3
4	7.350e9	1.693e9	**7.348e9**	**1.692e9**	7.348e9	1.692e9	7.348e9	1.692e9	7.350e9	1.693e9
5	**1.565e−3**	**3.522e−3**	4.454e−2	7.985e−2	3.440e−2	2.634e−2	3.344e0	1.261e0	1.879e1	2.956e0
6	**2.863e5**	**1.531e2**	2.877e5	6.196e2	2.874e5	1.891e2	2.871e5	3.489e2	2.901e5	2.353e2
7	**6.118e4**	**9.268e0**	6.181e4	2.635e2	6.154e4	6.462e1	6.163e4	1.322e2	6.337e4	1.303e2
8	**1.331e4**	**9.561e0**	1.437e4	4.110e2	1.386e4	1.835e2	1.405e4	2.755e2	1.683e4	2.239e2
9	**3.232e4**	**1.671e1**	3.419e4	5.558e2	3.318e4	2.231e2	3.372e4	3.316e2	3.772e4	2.846e2
10	**3.412e3**	**3.412e0**	3.923e3	1.848e2	3.606e3	5.379e1	3.635e3	1.636e2	5.315e3	1.306e2
11	**1.481e5**	**3.573e1**	1.611e5	3.651e3	1.543e5	1.603e3	1.583e5	2.310e3	1.839e5	1.222e3
12	**6.533e0**	**1.833e0**	1.140e1	3.701e0	1.033e1	2.578e0	1.317e1	3.869e0	1.503e1	3.996e0
13	6.900e2	1.106e1	6.633e2	7.417e0	**6.601e2**	**2.869e0**	6.659e2	6.209e0	7.077e2	7.173e0
14	**2.645e3**	**6.922e0**	3.141e3	8.570e1	2.884e3	7.197e1	3.070e3	6.617e1	3.838e3	3.599e1
15	**9.792e−3**	**6.460e−4**	1.247e−2	2.683e−3	8.007e−2	4.316e−2	1.957e−1	5.547e−2	2.302e−1	1.031e−1
16	**3.504e3**	**9.936e0**	3.915e3	1.866e2	3.672e3	7.427e1	3.870e3	1.135e2	5.081e3	5.324e1
17	**4.250e3**	**8.970e0**	4.758e3	1.774e2	4.415e3	6.056e1	4.625e3	1.006e2	5.967e3	5.800e1

4. Discussion

4.1. Algorithm performances

According to the training period results shown in Table 2, the SP-UCI-based ANN is able to reach the lowest final objective function values (the smallest errors) for 12 out of 17 datasets when compared to the results derived with other EAs-based ANN. According to the validation and testing results shown in Table 2, the SP-UCI-based ANN outperforms other algorithms-based ANNs for 11 out of 17 datasets. Furthermore, for the datasets that SP-UCI-based ANN did not perform as the best algorithm (datasets 1, 3, 4, 5, 13, and 14 of Table 2), the final objective function values (error terms) obtained by SP-UCI-enhanced ANN are rather comparable with the best performing model, respectively. According to the training datasets error trajectories shown in Fig. 2, generally, SP-UCI, PSO, and GA are able to find lower objective function values than SA and DE. For most of the datasets, the final objective function values reached by SP-UCI are slightly lower than those produced by PSO and GA.

Another interesting finding is that in some datasets, such as datasets 1, 2, 4, 5, 7, 9, and 13, SA and GA are surprisingly fast in optimizing the objective function values, especially during the beginning 20% of the entire evolutions. In these datasets, SP-UCI is relatively slow at the beginning of the evolutions, and the best function values during iterations are high. Nevertheless, the evolutions of SP-UCI are able to pursue low errors and surpass the performances of many other algorithms at the end of the search. This is because the optimization scheme used in SP-UCI is adaptive and self-regulated during the entire evolution. In the beginning of all runs, the search by SP-UCI focuses on a relatively large domain of parameter spaces, and the complexes are recursively shuffled in order to exchange information of the response surface, i.e., the current best fitness values. As the search continues, the constructed simplex in each complex becomes smaller due to the shrinking strategy used in the Nelder–Mead simplex downhill scheme. Thus, the search domain gradually transforms from a large-scale, global region into a smaller local area, where the search quickly converges. As a result, the best objective function values eventually found by the simplex-complex scheme in SP-UCI-based ANN are demonstrating the lowest against other EAs in a single run for most of the datasets as demonstrated in Table 2.

4.2. Consistency and reproductivity

In order to investigate the consistency and reproductivity of different EA-enhanced ANNs, we carried out two scenario tests. Under scenario 1, we carried out 30 independent runs on each dataset using the same training, validation, and testing datasets. Under scenario 2, another 30 runs were performed using different shuffled data. In other words, the experiments under scenario 2 use different training, validation, and testing datasets. The results for the training datasets for all 30 runs under scenario 1 and 2 are shown in Tables 3 and 4, respectively. Furthermore, the algorithm performances on testing datasets under scenario 1 and 2 are shown in Tables 5 and 6, respectively.

According to the training results for 30 independent runs (Table 3) and 30 shuffled runs (Table 4), the mean and Std of the final optimized objective function values with SP-UCI are consistently lower than those using other EA-based ANNs for 12 out of 17 datasets, and 11 out of 17 datasets, respectively. When compared with the single run results shown in Table 2, some slight differences among multiple runs are observed with regard to the best objective function values obtained by each algorithm. The differences among different run results under the same scenario indicate the randomness and deceptiveness

Table 6
The statistical performances of SP-UCI-, PSO-, GA-, SA-, and DE-based ANN on 30 randomly shuffled testing datasets (bold and underlined values indicate the best statistics for each dataset).

Dataset No.	SP-UCI		PSO		GA		SA		DE	
	Mean	Std	Mean	Std	Mean	Std	Mean	Std	Mean	Std
1	3.999e4	7.466e3	3.999e4	7.467e3	**3.998e4**	7.467e3	3.998e4	**7.466e3**	4.001e4	7.469e3
2	2.278e3	3.094e2	2.170e3	2.738e2	**2.153e3**	**2.723e2**	2.202e3	2.836e2	2.911e3	2.996e2
3	7.350e7	1.693e7	**7.348e7**	**1.692e7**	7.348e7	1.692e7	7.348e7	1.692e7	7.350e7	1.693e7
4	9.400e1	3.448e1	**9.540e0**	**4.293e0**	3.142e1	9.018e0	1.579e1	4.770e0	1.053e2	2.137e1
5	**3.079e−2**	**9.076e−2**	1.326e−1	2.275e−1	1.306e−1	1.176e−1	2.415e0	1.629e0	1.532e1	3.321e0
6	**2.868e6**	**4.896e3**	2.881e6	7.687e3	2.879e6	5.447e3	2.876e6	7.058e3	2.907e6	5.042e3
7	**1.059e5**	**8.693e4**	1.064e5	8.706e4	1.062e5	8.701e4	1.063e5	8.706e4	1.080e5	8.763e4
8	**1.405e4**	**2.004e3**	1.536e4	2.139e3	1.461e4	2.085e3	1.504e4	2.116e3	1.784e4	2.303e3
9	**3.359e4**	**4.136e3**	3.536e4	4.241e3	3.452e4	4.218e3	3.515e4	4.364e3	3.908e4	4.567e3
10	**2.669e3**	**4.976e2**	3.144e3	5.768e2	2.842e3	5.347e2	2.872e3	5.329e2	4.219e3	7.515e2
11	**1.606e5**	**1.128e4**	1.732e5	1.247e4	1.663e5	1.164e4	1.707e5	1.218e4	1.976e5	1.251e4
12	**4.167e0**	**3.364e0**	5.833e0	4.511e0	6.100e0	3.566e0	8.000e0	5.003e0	1.013e1	5.158e0
13	3.906e3	5.208e3	3.879e3	5.188e3	**3.871e3**	5.184e3	3.879e3	**5.183e3**	3.934e3	5.226e3
14	**2.244e3**	3.286e2	2.665e3	**3.146e2**	2.454e3	3.375e2	2.572e3	3.472e2	3.317e3	3.900e2
15	**1.426e−2**	**2.604e−3**	1.936e−2	7.741e−3	9.397e−2	9.266e−2	1.948e−1	8.239e−2	2.381e−1	9.637e−2
16	**3.537e3**	**3.384e2**	4.025e3	3.857e2	3.685e3	3.521e2	3.907e3	3.427e2	5.088e3	4.018e2
17	**4.171e3**	**2.956e2**	4.741e3	3.175e2	4.356e3	3.162e2	4.548e3	3.486e2	5.881e3	3.705e2

associated with each single individual run due to computation and rounding errors. According to the averaged statistics shown in Table 3, the mean and Std of final objective function values produced by SP-UCI-, PSO-, and GA-based ANNs are consistently lower than those produced by SA- and DE-based ANNs. A lower value on mean and Std indicates a better convergence. The averaged algorithm performances can be ranked as SP-UCI $>$ GA $>$ PSO $>$ SA $>$ DE, if sorely based on the averaged best objective function values shown in Table 3. Similarly, based on the averaged performances of 30 runs conducted on the shuffled training datasets (Table 4), the algorithm performance ranking is SP-UCI $>$ PSO $>$ GA $>$ SA $\approx$ DE.

Despite the algorithm performances on training datasets, more importantly, the trained ANN weights are evaluated on testing datasets to show the accuracy of model performances in practice. According to the testing results for 30 independent runs (Table 5), SP-UCI-enhanced ANN is able to produce the lowest mean and Std values for 12 out of 17 datasets, which suggests a superior convergence capability of SP-UCI-based ANN over other EA-enhanced ANNs. In addition, according to the testing performances using the trained weights from 30 shuffled datasets (Table 6), SP-UCI-enhanced ANN is able to reach the lowest mean and Std values for 11 out of 17 datasets when compared to those with others. As shown in Tables 5 and 6, PSO-, GA-, SA-, and DE-based ANNs are able to reach the lowest mean and Std values in a few testing datasets individually. However, the lowest mean and Std values are not achievable at the same time. In other words, PSO-, GA-, SA-, and DE-based ANNs sometimes can obtain the lowest mean objective function value, but the variability of the final convergence is relatively large when repeating multiple runs. On the contrary, an algorithm with a relatively small Std value for multiple runs does not necessary generate the lowest errors in the testing phase. In the datasets in which SP-UCI-based ANN outperforms other algorithm, the lowest mean and Std values are achievable simultaneously. This means that the performance of the SP-UCI-enhanced ANN shows a consistently good convergence during the testing phase. Based on the statistics of the final objective function values provided in Tables 5 and 6, generally, the algorithm performances on testing datasets rank as follows: SP-UCI $>$ GA $>$ PSO $>$ SA $>$ DE.

4.3. Uncertainties and Trade-offs

There are two main reasons for the different convergence performances using different kinds of EAs-enhanced ANNs. One aspect is solely related to the search mechanism itself, i.e., the operators and logic used in various EAs. Some algorithms are efficient for global optimization and some are good for local search. Furthermore, the size of a population occasionally turns out to be sensitive and influential on the final convergence. With a larger population, the chances of finding a better global optimum will increase for all EAs. This is not only because the initial sampling will cover a larger parameter domain, but also because the efficiency of information exchange on the best fitness found so far will be increased during the entire evolution. Another uncertainty source is due to the different shapes of response surfaces that are associated with various problems, i.e., the flatness and roughness of the objective function space. In some of the benchmark databases, such as databases 1, 3, and 6, the objective function space is relatively flatter than others. In other words, the difference between the objective function values in global optimum and any randomly sampled position is small, which destructively creates many large and flat valley-shaped search domains. The complexity of response surfaces, either the large flat valley-shape area, or the rough area with many local minimums, will introduce huge challenges to the global convergence of each EA. For example, with regard to databases 1, 3, and 6, the magnitudes of objective function values are very large, and the difference between initial sampling and final minimized objective function values is relatively small. In the training phases on these databases, the evolution processes of all types of EAs are inevitably under the risks of pre-convergence. When producing the

next generation, no single algorithm has the proper mechanism to guarantee the discovery of the global optimum, which is possibly located in a far-away position with only relatively small variance as compared to the current best fitness in the search domain.

It is worth mentioning that the demonstrated comparison among different EAs is subject to the famous No-Free-Lunch (NFL) theorems [43]. According to the NFL theorem by Wolpert and Macready [43], the enhanced performances in any algorithm on a set of problems is always offset by another set of problems. Here, the performances refer to all possible measures that are associated with the algorithm, such as effectiveness, efficiency, uncertainty, flexibility, reproductivity, and suitability, etc. In the practical use of EAs, the effectiveness (convergence) and computational efficiency (speed) belong to two of the crucial measures that users mostly concern. In our experiments, there is a conspicuous tradeoff between convergence and speed. For example, SA- and GA-enhanced ANNs tend to have superior computational efficiency at the beginning of the evolutions, such as on databases 1, 2, 4, 5, 9, 10, 11, 13, 14, and 16. However, the greedy search mechanisms used in SA and GA cannot generate the lowest final converged objective function values when compared to other EAs. In other words, the search schemes in SA and GA are not able to guarantee the global convergence. In contrast, according to Fig. 2, SP-UCI- and PSO-based ANN are much slower than the SA- and GA-based ANN during the beginning of evolutions. However, lower final objective function values, or a better convergence, are achievable on most of the databases using SP-UCI- and PSO-enhanced ANN. A rational explanation is that during the beginning of the evaluation, search strategies used in SP-UCI and PSO are more rigorously progressive, i.e., algorithm tends to do more comprehensive and detailed search instead of forcing the population toward the fastest gradient decreasing direction, than those used in other EAs. The strategic slowdown in the searches by SP-UCI and PSO during the beginning of evolutions allows the algorithms to frequently reckon the best searching directions towards the global optimum and to efficiently infer the position of global optimums in the parameter space. The risks of population being trapped in deceptive local optimums can be reduced in SP-UCI and PSO at the beginning of search.

5. Conclusions

In this paper, a newly developed SP-UCI-enhanced ANN is presented. The SP-UCI algorithm is an efficient and effective global evolutionary optimization scheme, which has never been employed nor tested in the field of tuning ANN connectivity weights. The performance of SP-UCI-enhanced ANN is proven to be overwhelming, or at least competitive, to other commonly used EA-based ANN, including PSO, GA, SA, and DE. The following conclusions are drawn based on the presented experiments:

(1) The SP-UCI algorithm is proven to be an efficient and effective EA with regard to producing optimized ANN connectivity weights on most of the tested datasets.
(2) During the beginning of the evolution, SP-UCI and PSO are less efficient than others, i.e., SA and GA. However, the final convergence turns out to be competitive at the end of the search. The different performances are due to the differences among the searching mechanisms used in various EAs, such as crossover, mutation, and shrinking of simplex, etc.
(3) The use of EAs in the field of ANN design includes multiple aspects, such as connectivity weight training, ANN structure, and topography. Weight-space training is as important as the optimal design of the ANN structure and topography.

Future works are recommended to investigate the performances of using SP-UCI-, GA-, PSO-, SA-, and DE-enhanced ANN structure and topography designing. To the authors' knowledge, an attempt to use SP-UCI in optimizing ANN's structure and topography has never been reported in the literature, as well as the use in training ANN connectivity weights. As demonstrated in this paper, the SP-UCI algorithm is a powerful tool in tuning ANN weights, and its usefulness in optimally designing the ANN structure is worth investigating. A recent paper by Zhang et al. [50] also introduced a novel way of solving optimization tasks by Neural Network. Furthermore, some recent studies conducted by Yang et al. [47] pointed out that the performances of the ANN combined with BP and a gradient-descent scheme are limited with regard to monthly and seasonal streamflow prediction. They suggested that the ANN training with heuristic search optimization schemes can be tested further, especially for the chaotic natural systems, and the complex human decision making processes. Authors also suggest future work could be carried out to test the SP-UCI-ANN framework on other high-nonlinear problems, such as the rainfall-runoff processes [27] and probability distribution estimates [26]. Furthermore, different EA-enhanced ANN training can be tested on many recent developed Recurrent Neural Network and Convolutional Neural Network for solving temporal and spatial classification and regression problems.

Acknowledgments

The financial support of this research are from U.S. Department of Energy (DOE Prime Award # DE-IA0000018), California Energy Commission (CEC Award # 300-15-005), MASEEH fellowship, NSF CyberSEES Project (Award CCF-1331915), NOAA/NESDIS/NCDC (Prime award NA09NES4400006 and NCSU CICS and subaward 2009-1380-01), and the U.S. Army Research Office (award W911NF-11-1-0422). The authors would like to use this study as a dedication to our co-author Prof. Xiaogang Gao for his scientific contributions to optimization, automatic model calibration, and water resources. Prof. Xiaogang Gao recently retired after over three decades of a successful and productive academic career. He was one of the

main contributors and original developers for the SCE-UA algorithm and its families, including the Multi-Objective Complex Evolution Algorithm (MOCOM-UA), the Shuffled Complex Evolutionary Algorithm with Principal Component Analysis (SP-UCI), and the Multi-Objective Complex Evolution Algorithm with Principal Component Analysis and Crowding Distance (MOSPD-UCI).

References

[1] A. Cichocki, S.-I. Amari, Adaptive Blind Signal and Image Processing: Learning Algorithms and Applications, 1, John Wiley & Sons, 2002.
[2] W. Chu, X. Gao, S. Sorooshian, A new evolutionary search strategy for global optimization of high-dimensional problems, Inf. Sci. 181 (2011) 4909–4927.
[3] W. Chu, T. Yang, X. Gao, Comment on "High-dimensional posterior exploration of hydrologic models using multiple-try DREAM (ZS) and high-performance computing, in: Eric Laloy, Jasper A. Vrugt (Eds.), Water Resources Research, 50, 2014, pp. 2775–2780.
[4] C. Dai, W. Chen, Y. Zhu, Z. Jiang, Z. You, Seeker optimization algorithm for tuning the structure and parameters of neural networks, Neurocomputing 74 (2011) 876–883.
[5] S. Ding, H. Li, C. Su, J. Yu, F. Jin, Evolutionary artificial neural networks: a review, Artif. Intel. Rev. 39 (2013) 251–260.
[6] Q. Duan, S. Sorooshian, V. Gupta, Effective and efficient global optimization for conceptual rainfall-runoff models, Water Resour Res 28 (1992) 1015–1031.
[7] R.C. Eberhart, J. Kennedy, A new optimizer using particle swarm theory, in: Proceedings of the Sixth International Symposium on Micro Machine and Human Science, 1995, pp. 39–43.
[8] R. Eglese, Simulated annealing: a tool for operational research, Eur. J. Oper. Res. 46 (1990) 271–281.
[9] D.B. Fogel, Evolutionary Computation: Toward a New Philosophy of Machine Intelligence, 1, John Wiley & Sons, 2006.
[10] D.E. Golberg, in: Genetic Algorithms in search, Optimization, and Machine Learning, Addion Wesley, 1989, p. 102. 1989.
[11] V. Granville, M. Krivánek, J.-P. Rasson, Simulated annealing: A proof of convergence, IEEE Trans. Pattern Anal. Mach. Intel. 16 (1994) 652–656.
[12] V.G. Gudise, G.K. Venayagamoorthy, Comparison of particle swarm optimization and backpropagation as training algorithms for neural networks, in: Proceedings of the 2003 IEEE Swarm Intelligence Symposium, 2003, pp. 110–117.
[13] H.V. Gupta, H. Kuo-lin, S. Sorooshian, Superior training of artificial neural networks using weight-space partitioning, in: Proceedings of the International Conference on Neural Networks, 3, 1997, pp. 1919–1923.
[14] Y. Hirose, K. Yamashita, S. Hijiya, Back-propagation algorithm which varies the number of hidden units, Neural Netw 4 (1991) 61–66.
[15] J. Ilonen, J.-K. Kamarainen, J. Lampinen, Differential evolution training algorithm for feed-forward neural networks, Neural Process. Lett. 17 (2003) 93–105.
[16] A.K. Jain, R.P.W. Duin, J. Mao, Statistical pattern recognition: A review, IEEE Trans. Pattern Anal. Mach. Intell. 22 (2000) 4–37.
[17] A. Jain, S. Srinivasulu, Development of effective and efficient rainfall-runoff models using integration of deterministic, real-coded genetic algorithms and artificial neural network techniques, Water Resourc. Res. 40 (2004).
[18] E.M. Johansson, F.U. Dowla, D.M. Goodman, Backpropagation learning for multilayer feed-forward neural networks using the conjugate gradient method, Int. J. Neural Syst. 2 (1991) 291–301.
[19] I. Jolliffe, Principal Component Analysis, Wiley Online Library, 2002.
[20] J. Kennedy, J.F. Kennedy, R.C. Eberhart, Y. Shi, Swarm Intelligence, Morgan Kaufmann, 2001.
[21] J. Kennedy, Particle swarm optimization, in: Encyclopedia of Machine Learning, Springer, 2011, pp. 760–766.
[22] S. Kirkpatrick, Optimization by simulated annealing: Quantitative studies, J. Stat. Phys. 34 (1984) 975–986.
[23] S. Lek, J.F. Guegan, Artificial neural networks as a tool in ecological modelling, an introduction, Ecol. Model. 120 (1999) 65–73 Aug 17.
[24] M. Lichman, UCI Machine Learning Repository, 2013.
[25] R. Lippmann, An introduction to computing with neural nets, IEEE ASSP Magazine 4 (1987) 4–22.
[26] X. Liu, Y. Luo, T. Yang, K. Liang, M. Zhang, C. Liu, Investigation of the probability of concurrent drought events between the water source and destination regions of China's water diversion project, Geophys. Res. Lett. 42 (20) (2015) 8424–8431.
[27] X. Liu, T. Yang, K. Hsu, C Liu, S. Sorooshian, Evaluating the streamflow simulation capability of PERSIANN-CDR daily rainfall products in two river basins on the Tibetan Plateau, Hydrol. Earth Syst. Sci. 21 (1) (2017) 169–181.
[28] H.R. Maier, G.C. Dandy, Neural networks for the prediction and forecasting of water resources variables: a review of modelling issues and applications, Environ. Model. Softw. 15 (2000) 101–124.
[29] W.S. McCulloch, W. Pitts, A logical calculus of the ideas immanent in nervous activity, Bull. Math. Biophys. 5 (1943) 115–133.
[30] J.A. Nelder, R. Mead, A simplex method for function minimization, Comput. J. 7 (1965) 308–313.
[31] M.A. Panduro, C.A. Brizuela, L.I. Balderas, D.A. Acosta, A comparison of genetic algorithms, particle swarm optimization and the differential evolution method for the design of scannable circular antenna arrays, Progr. Electromag. Res. B 13 (2009) 171–186.
[32] L. Poikolainen, F. Neri, F. Caraffini, Cluster-based population initialization for differential evolution frameworks, Inform. Sci. 297 (2015) 216–235.
[33] L.Y. Pratt, Comparing Biases For Minimal Network Construction with Back-Propagation, 1, Morgan Kaufmann Pub, 1989.
[34] K. Price, R.M. Storn, J.A. Lampinen, Differential Evolution: A Practical Approach to Global Optimization, Springer Science & Business Media, 2006.
[35] D.E. Rumelhart, G.E. Hinton, and R.J. Williams, Learning Internal Representations by Error Propagation, DTIC Document1985.
[36] A.R. Simpson, G.C. Dandy, L.J. Murphy, Genetic algorithms compared to other techniques for pipe optimization, J. Water Resour. Plann. Manage. 120 (1994) 423–443.
[37] A. Slowik, M. Bialko, Training of artificial neural networks using differential evolution algorithm, in: Proceedings of the 2008 Conference on Human System Interactions, 2008, pp. 60–65.
[38] R. Storn, K. Price, Differential evolution–a simple and efficient heuristic for global optimization over continuous spaces, J. Global Optim. 11 (1997) 341–359.
[39] M.D. Sulistiyo, R.N. Dayawati, Evolution strategies for weight optimization of Artificial Neural Network in time series prediction, in: Proceedings of the 2013 IEEE International Conference on Robotics, Biomimetics, and Intelligent Computational Systems (ROBIONETICS), 2013, pp. 143–147.
[40] R.S. Sutton, Two problems with backpropagation and other steepest-descent learning procedures for networks, in: Proceedings of the 8th Annual Conference Cognitive Science Society, 1986, pp. 823–831.
[41] P. Werbos, Beyond regression: New tools for prediction and analysis in the behavioral sciences, Doctoral Dissertation, Applied Mathematics, Harvard University, MA, 1974.
[42] D. Whitley, T. Starkweather, C. Bogart, Genetic algorithms and neural networks: Optimizing connections and connectivity, Parallel Comput. 14 (1990) 347–361.
[43] D.H. Wolpert, W.G. Macready, No free lunch theorems for optimization, IEEE Trans. Evol. Comput. 1 (1997) 67–82.
[44] B.J. Wythoff, Backpropagation neural networks: a tutorial, Chemom. Intell. Lab. Syst. 18 (1993) 115–155.
[45] X. Yao, Evolving artificial neural networks, Proc. IEEE 87 (1999) 1423–1447.
[46] T. Yang, X. Gao, S.L. Sellars, S. Sorooshian, Improving the multi-objective evolutionary optimization algorithm for hydropower reservoir operations in the California Oroville–Thermalito complex, Environm. Model. Softw. 69 (2015) 262–279.
[47] T. Yang, A.A. Asanjan, A.E. Welles, X. Gao, S. Sorooshian, and X. Liu, Developing reservoir monthly inflow forecasts using artificial intelligence and climate phenomenon information. Water Resour. Res., 53, 2017, 2786–2812 doi:10.1002/2017WR020482.

[48] T. Yang, X. Gao, S. Sorooshian, and X. Li, Simulating California reservoir operation using the classification and regression-tree algorithm combined with a shuffled cross-validation scheme, Water Resour Res, 52, 2016, 1626–1651, doi:10.1002/2015WR017394.
[49] T. Yang, Y. Tao, J. Li, Q. Zhu, L. Su, X. He, X. Zhang, Multi-criterion model ensemble of CMIP5 surface air temperature over China, Theor. Appl. Climatol. (2017) 1–16.
[50] G. Zhang, H. Rong, F. Neri, M.J. Pérez-Jiménez, An optimization spiking neural P system for approximately solving combinatorial optimization problems, Int. J. Neural Syst. 24 (05) (2014) 1440006.

Tiantian Yang was born in Beijing, China, in 1987. He received the B.S. and M.S. degree in mechanical engineering from Tsinghua University, Beijing, China in 2009, and from University of California, Irvine, in 2010, respectively. In 2015, he received his Ph.D. degree in civil engineering from University of California, Irvine with focus on Civil Engineering and water resources. From 2015 to present, he is a project scientist in the Henry Samueli School of Engineering at University of California-Irvine, Irvine, CA92687, U.S.A., and a research scientist at Deltares USA Inc, Silver Spring, Maryland, USA. Dr. Tiantian Yang's research focuses on optimization, artificial neural network, evolutionary computation, machine learning, data-mining, reservoir decision making, hydropower operation, water resources management, and climate change.

Ata Akbari Asanjan was born in Tabriz, Iran in 1991. He received the B.S. in civil engineering-water resources from University of Tabriz, Iran in 2014 and the M.S. degree in civil engineering-water resources from University of California Irvine, Irvine, CA in 2016. He is currently pursuing Ph.D. in civil engineering-water resources in University of California-Irvine. His main research interests are sequence learning and machine learning for classification/regression in hydrometeorology.

Mohammad Faridzad was born in Tehran, Iran, in 1988. He received the B.S. degree in civil engineering from AmirKabir University of Technology, in 2011 and the M.S. degree in environmental engineering from Sharif University of Technology, Tehran, Iran, in 2013. He is currently a research assistant and pursuing his Ph.D. degree in Hydrology and water resources engineering at the Center for Hydrometeorolog and Remote Sensing(CHRS) at University of California, Irvine. Mr. Faridzad is a member of American Geophysical Union(AGU) and was the recipient of MASEEH fellowship for his Ph.D. studies in 2015.

Negin Hayatbini was born in Tehran, Iran, in 1990. She received her B.S. degree in civil engineering from National Zanjan University, Zanjan, Iran, in 2011 and the M.S. degree in civil engineering in Hydraulic Structures major from Sharif University of technology, Tehran, Iran, in 2014. She is currently pursuing her Ph.D. degree in Civil engineering, Hydrology major at University of California, Irvine. Ms. Hayatbini is member of American Geophysical Union (AGU) since 2014, and is the recipient of the MASEEH fellowship from the University of California, Irvine.

Xiaogang Gao (retired) was born in Shanghai, China. He received the Ph.D. degree from the Department of Hydrology and Water Resources, University of Arizona, Tucson, in 1993. Since 2003, he has been a Professor in the Department of Civil and Environmental Engineering, University of California, Irvine. His research interests include hydrology and water resource management, coupled atmosphere-land surface modeling, optimization, and statistical data assimilation. Dr. Xiaogang Gao was a member of American Geophysical Union (AGU), a member of American Meteorology Society (AMS), and a member of American Society of Civil and Engineers (ASCE).

Soroosh Sorooshian was born in Kerman, Iran, in 1949. He received a B.S. in Mechanical Engineering from California State Polytechnic University. He received a M.S., an Engineer Degree, and a Ph.D. from University of California, Los Angeles, focusing on Operations Research, Systems Engineering and Engineering, respectively. Dr. Soroosh Sorooshian's expertise and the research focuses of his team include artificial intelligence, optimization, hydrometeorology, water resources systems, hydropower, climate studies and application of artificial intelligence techniques to earth science problems with special focus on the hydrologic cycle and water resources issues of arid and semi-arid zones.